Python & In-memory CGNS trees

Using CGNS trees for Code-coupling
Code life cycle

▷ Idea/Code/Test/Change
  - Prototype
  - Test
  - Pre/Post processing
  - Code-coupling
  - Parallel

▷ All you can do with another programming language
  - Interpreted
  - Actually dedicated to code gluing
  - Script languages are easily extensible

▷ Baseline for an Open System
Python

▷ Object-oriented interpreted language
  ▶ Very easy to learn
  ▶ Clear syntax
  ▶ Powerful numerical extensions
    ● Python/C/C++/Fortran arrays

▷ Good candidate for code gluing
  ▶ Pre & post processing on CGNS data
  ▶ A scripting language
▷ Python wrapper on CGNS MLL and ADF
  ▶ Straightforward mapping
  ▶ Use 100% python types
    ● Lists, strings, integers, floats
    ● Numerical array
      – Contiguous C/Fortran array
      – Points to actual memory zone

▷ Easy scripting
  ▶ Perform CGNS calls on-the-fly
Tree representation

- List of nodes
- Each node has...
  - A Name
  - A Type
  - A Value
  - A list of sons

Generic CGNS low level node requirements (ADF/HDF5)

```
[ 'Transform', (1, 2, 3), [], 'int[IndexDimension]' ],
[ 'PointRange', ((1, 1, 1), (1, 9, 9)), [], 'IndexRange_t' ],
[ 'PointRangeDonor', ((21, 1, 1), (21, 9, 9)), [], 'IndexRange_t' ]
```
File and memory

ADF/HDF5 file
- open/read/write/close
- MLL keeps private tree structure in memory
- ADF is per-node but still private data structure
  ▶ PyCGNS only maps to this behaviour

Python tree
- The Python/CGNS tree is just another implementation
- Structure in memory but not a proprietary one
  ▶ Same interface/Different implementation
The LOGICAL data model is unchanged: SIDS
```python
import CGNS
import numarray as N

x=y=z=N.zeros((3,5,7),'d')

a=CGNS.pyCGNS("newfile.cgns",CGNS.MODE_WRITE)

print a.error

idb=a.basewrite("Base",3,3)
idz=a.zonewrite(idb,"Zone 01",[3,5,7],CGNS.Structured)

a.coordwrite(idb,idz,CGNS.RealDouble,CGNS.CoordinateX,x)
a.coordwrite(idb,idz,CGNS.RealDouble,CGNS.CoordinateY,y)
a.coordwrite(idb,idz,CGNS.RealDouble,CGNS.CoordinateZ,z)

a.close()
```
Can I do this and that with CGNS?

- Just try it!
- Versatile testing support

```python
import CGNS

f = CGNS.pyCGNS("hydro-result.cgns", CGNS.MODE_WRITE)

f.basewrite("MASS2", 3, 3)
f.zonewrite(1, "Block01", (2, 3, 4, 1, 2, 3, 0, 0, 0), CGNS.Structured)
f.solwrite(1, 1, "07-01-1944 06:00:00", CGNS.CellCenter)
f.fieldwrite(1, 1, 1, CGNS.RealDouble, "sediment", w)
f.goto(1, [(CGNS.Zone_t, 1), (CGNS.FlowSolution_t, 1), (CGNS.DataArray_t, 1)])
f.descriptorwrite("Description", "Text here")
f.descriptorwrite("Units", "Text here")

f.close()
```
Scripting example: post-processing

▷ Add links to actual grids
   - The computation sessions results are sharing the same grid
   - No duplicates
   - Post-processing adds links to the actual grid
   - True MLL/ADF calls performed on file

```python
from CGNS import *

a=pyCGNS("result-001.cgns",MODE_MODIFY)

a.goto(1,[(Zone_t,1)])
a.linkwrite("GridCoordinates","grid.cgns","/Base/Zone/GridCoordinates")

a.close()
```
Structured grid seen as unstructured

- Generates connectivity
- Read the file/Change in-memory tree/Send to code
Code-coupling

▷ Blind connection to peer code
  ▶ Open System: Public interface
    - Common baseline
    - Restriction input/output
  ▶ Use Bct for data exchange
    - Input/Output: BCdataset
    - « Contact surface »
    - Strong requirements for an arbitrary exchange mean

▷ Efficiency
  - Memory +no data duplication
  - Easy stub & proto
Code-coupling CGNS tree

Python & In-memory CGNS trees

Marc Poinot – ONERA/DSNA
AIAA-SF-2006/CGNS-Tutorial
import MpCCI

pathB = "/FlatPlate/Fluid/ZoneBC/Wall:Heat/DataSet#01/NeumannData"
pathI = pathB + "/Temperature"
pathO = pathB + "/NormalHeatFlux"
it = E.iteration()

fqx = mcci.Parameter_info("Simulation_Fluid_2_Therm_Ratio", MpCCI.CCI_INT)

xp = xw.get(E.RUNTIME_TREE)
xf = X.retrieve(pathO, xp)
if (xf and ((it % fqx) == 0)):
    sdl = mcci.Parameter_info("Fluid_Private_Synchro_ID", MpCCI.CCI_INT)
    ZID = mcci.Parameter_info("Global_Mesh_ID", MpCCI.CCI_INT)
    BID = 1
    nnodes = len(xf[1].flat)
    if (it % fqx) == 0:
        mcci.Put_nodes(ZID, BID, 171, 1, nnodes, 0, None, MpCCI.CCI_DOUBLE, xf)
        mcci.Reach_sync_point(sdl)

(rC, nC) = mcci.Get_nodes(ZoneID, BoundaryID, 154, 1, nnodes, 0, None, MpCCI.CCI_DOUBLE)
...
E.update((E.RUNTIME_TREE, rt)
import elsApy as E
from Scientific import MPI

communicator=MPI.world.duplicate()
id = communicator.rank
if ( id == 0 ): remoteId=1
elif ( id == 1 ): remoteId=0

datatree=E.get(E.RUNTIME_TREE)
temp=pickle.dumps(datatree)
communicator.nonblocking_send(temp, remoteId, id)
return,rank,tag=communicator.receiveString(None,None)
result=pickle.loads(return)

for l in result:
    if (l[0] == "RunTimeTree"):
        for ll in l[2]:
            if (ll[0] == "Rotor#Output"): ll[0]="Stator#Input"
            if (ll[0] == "Stator#Output"): ll[0]="Rotor#Input"

E.update(E.RUNTIME_TREE,result)
In-memory issues

▷ Dedicated to a platform
  ▶ One per platform: requires an API
  ▶ Translation mandatory between platforms
    ● XDR-like

▷ Best should be
  ▶ Use an existing system
    ● Python/Numeric (+Marshalling)
    ● HDF5 (?)
List of Python objects

- MLL-like interface
  - NewBase
  - NewZone
  - NewGridCoordinates
  - NewCoordinates
  - NewDataArray

- Numeric Python arrays
- Input/Output from MLL
- Use paths instead of ids
  - GetByExactPath
  - GetByRegexpPath
  - GetAllTreePath

```python
T=CGNSTree()
base=newBase(T,"Base",3,3)
print T
getChildrenNameByPath(T,"/Base/Zone-002/GridCoordinates")
```

```
[['CGNSLibraryVersion', 2.4, [], 'CGNSLibraryVersion_t'], ['Base', array([3, 3]), [], 'CGNSBase_t']]
```
Script example: Python/CGNS tree

```python
T=C.newCGNS()

base=C.newBase(T,"Base",3,3)
size=(20,10,5)

z1=C.newZone(base,"Zone-001",size)
C.newCoordinates(z1,"CoordinatesX",x)
C.newCoordinates(z1,"CoordinatesY",y)

f=open("T01.py","w+")
f.write(str(T))
f.close()

clist=C.getChildrenNameByPath(T,"/Base/Zone-002/GridCoordinates")
for c in clist:
    n=C.getByExactPath(T,"/Base/Zone-002/GridCoordinates/"+c)
    print C.nodeName(n)
    v=C.nodeValue(n)

print C.getChildrenType(T,"CGNSBase_t")
print C.getAllTreePath(T)
print C.getAllTreeType(T,"Zone_t")
print C.getAllTreeType(T,"DataArray_t")
```
Use tools operating on data trees
   - A data model is described by a grammar: SIDS
   - Translate the grammar for existing tools
     - Relax-NG, BNF, ...

In-Memory data structure can be used for...
   - Perform tree verification
   - Operate tree as ADT
     - Generate code:
       - MLL/ADF/HDF5/XML/SQL/XDR/...
CGNS is more than a storage mean...

- **CGNS as a data model**
  - Store data the « CGNS way »
    - e.g. Map to 100% python objects
  - Tree with public definition

- **CGNS as component interface**
  - Code-coupling data model
  - Transfer whole tree instead of arrays
    - e.g. Memory buffer based system